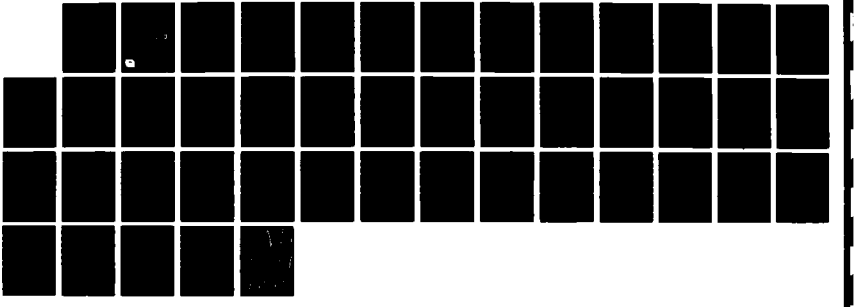
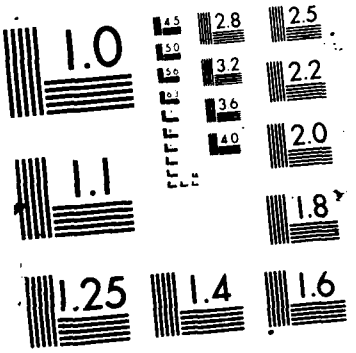
AD-A185 665 NEED AND RATIONALE FOR THE SOFTWARE TECHNOLOGY FOR ADAPTABLE RELIABLE SYS..(U) INSTITUTE FOR DEFENSE ANALYSES ALEXANDRIA VA S T REDWINE ET AL. SEP 85 1/1

UNCLASSIFIED IDA-P-1872 IDA/HQ-85-29683 MDA903-84-C-0031 F/G 12/5 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

IDA PAPER P-1872

NEED AND RATIONALE FOR THE SOFTWARE TECHNOLOGY FOR ADAPTABLE RELIABLE SYSTEMS (STARS) PROGRAM

Samuel T. Redwine, Jr.
Sarah H. Nash

September 1985

Prepared for

Office of the Under Secretary of Defense for Research and Engineering

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, VA 22311

IDA Log No. HQ 85-29683

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, or (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Papers

Papers normally address relatively restricted technical or policy issues. They communicate the results of special analyses, interim reports or phases of a task, ad hoc or quick reaction work. Papers are reviewed to ensure that they meet standards similar to those expected of refereed papers in professional journals.

Memorandum Reports

IDA Memorandum Reports are used for the convenience of the sponsors or the analysts to record substantive work done in quick reaction studies and major interactive technical support activities; to make available preliminary and tentative results of analyses or of working group and panel activities; to forward information that is essentially unanalyzed and unevaluated; or to make a record of conferences, meetings, or briefings, or of data developed in the course of an investigation. Review of Memorandum Reports is suited to their content and intended use.

The results of IDA work are also conveyed by briefings and informal memoranda to sponsors and others designated by the sponsors, when appropriate.

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

This paper has been reviewed by IDA to assure that it meets high standards of thoroughness, objectivity, and sound analytical methodology and that the conclusions stem from the methodology.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY DD Form 254 Dtd 1 October 1983	3. DISTRIBUTION/AVAILABILITY OF REPORT Public release/unlimited distribution.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) IDA Paper P-1872	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Institute for Defense Analyses 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION DoD-IDA Management Office (OUSDRE)
6c. ADDRESS (City, State, and ZIP Code) 1801 N. Beauregard Street Alexandria, VA 22311	7b. ADDRESS (City, State, and ZIP Code) 1801 N. Beauregard Street Alexandria, VA 22311
8a. NAME OF FUNDING/SPONSORING ORGANIZATION OUSDRE (R&AT) STARS Joint P.O. 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA 903 84 C 0031
8c. ADDRESS (City, State, and ZIP Code) 1211 Fern Street Arlington, VA 22202	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. T-4-236 / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Need and Rationale for the Software Technology for Adaptable, Reliable Systems (STARS) Program (U)

12. PERSONAL AUTHOR(S)
Samuel T. Redwine, Sarah H. Nash

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
FINAL	FROM ____ TO ____	September 1985	35

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
computer programming; computer program reliability; military requirements; long range (time); technology transfer software technology; weapon systems; computer personnel; STAR

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

There is evidence that the future software state of practice will have to plan and build many larger, more complex, more reliable, and more maintainable systems less labor-intensively than the current state of practice does today. Moreover, the current state of practice, in many cases, is having trouble meeting current requirements. Often, state-of-the-art technologies that could be used are not because widespread popularization of a software technology can take more than 15 to 20 years. Accelerating the transition of the state-of-the-art into the state-of-practice is one opportunity for closing this technology insertion gap. Where state-of-the-art technologies do not exist to improve the state-of-practice, R&D must be fostered.

The STARS (Software Technology for Adaptable Reliable Systems) program is necessary to accelerate, coordinate, and disseminate the results of R&D in software technology, bridging the gulf between future and current software technology states of practice, and meeting the need for an improved software state-of-practice.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ■ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code) / 22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE

IDA PAPER P-1872

NEED AND RATIONALE FOR THE SOFTWARE TECHNOLOGY FOR ADAPTABLE RELIABLE SYSTEMS (STARS) PROGRAM

Samuel T. Redwine, Jr.
Sarah H. Nash

September 1985

IDA

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-4-236

Acknowledgments

The authors are indebted to a number of people for their help on this paper. Vance Mall drafted an earlier document that served as a basis to this one and many of his thoughts and words from that document are incorporated here. Richard DeMillo, John Manley, and William Riddle helped to identify many of the sources cited in this report. They also reviewed the first edition of this report (IDA Memorandum Report M-57) and made many constructive suggestions. Special thanks go to Joyce Walker for typing the figures.

Table of Contents

1.0 Introduction

This document briefly presents the major arguments supporting the need for the Software Technology for Adapatable, Reliable Systems (STARS) program. The STARS program, part of the Department of Defense's Software Initiative, covers all aspects of the software life cycle from both technical and management viewpoints. It is intended to provide better management practices, improve software acquisition strategies, improve the underlying software technologies, increase personnel skill levels, create more powerful development and maintenance tools, increase the extent to which tools are used, and make advances in both software system methodology and software theory (120). Figure 1 is a graphic overview of the arguments for STARS and their relationships to one another.

Each section opens with a summary of the major points of the argument addressed in the section. A brief discussion of the argument with citations to the evidence that supports the argument follows. Rather than interrupt paragraphs by citing references at the end of each sentence, they are often grouped at the end of the discussion of a particular argument. Other references to articles and documents about STARS include (90)-(112), (137)-(140).

2.0 Future MCCR System Requirements

Future requirements for software within the DoD will be so severe that without improvements in software technology, the DoD will be unable to meet future needs. The points of this future requirements argument are summarized as follows. Software is becoming an increasingly important component of mission critical computer resource (MCCR) systems. The number, complexity, and criticality of functions performed by software in such systems are all increasing. Furthermore, there is a growing manpower shortage in the computer field that will compound future requirements unless ways are found to increase the productivity of computer personnel.

2.1 Software Requirements

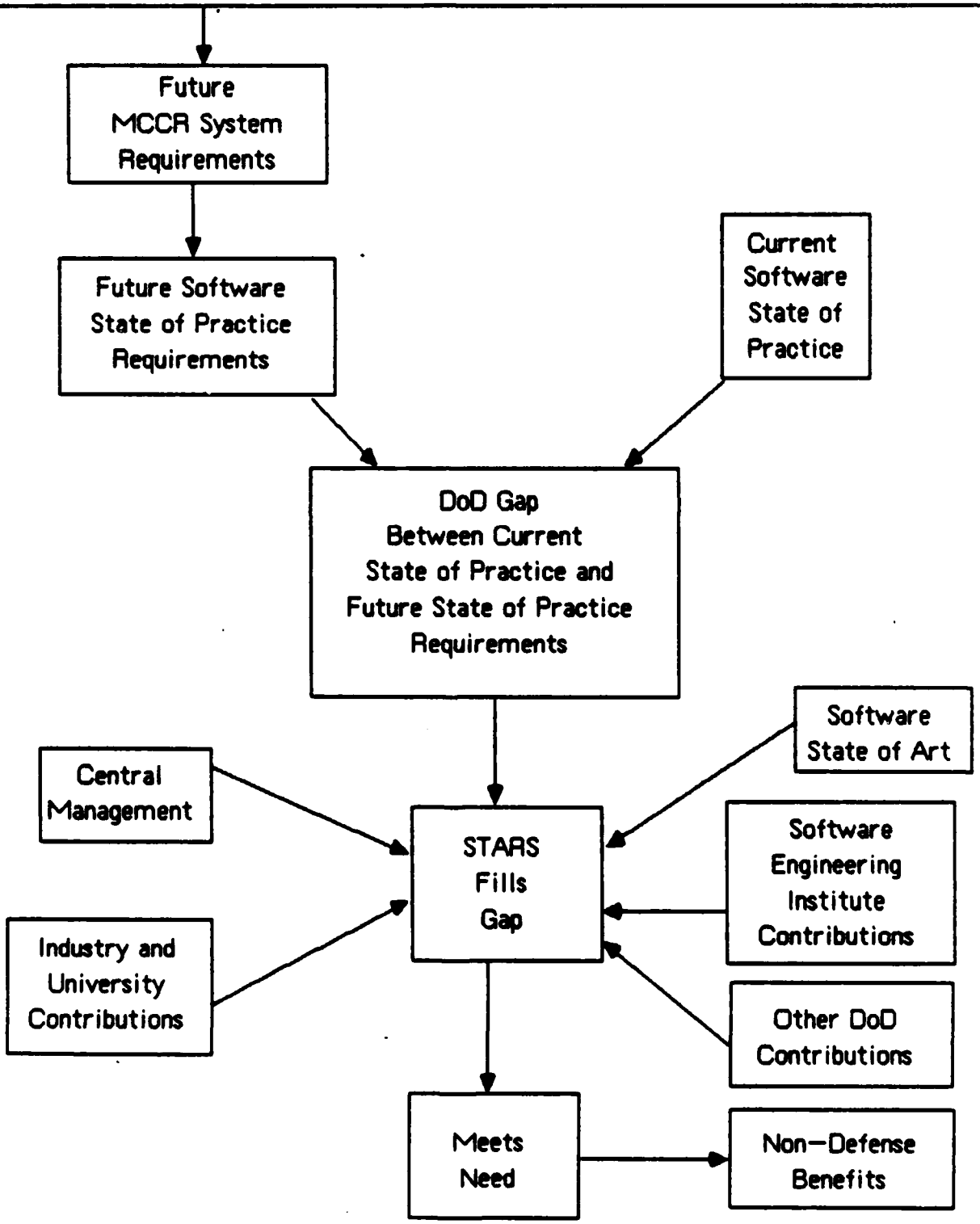
The last decade has seen a trend toward greater use of computers in virtually all weapon and support systems. Almost every item of defense equipment has one or more computer subsystem which controls its operation in some mission-critical fashion. From the avionics suite of the F/A-18 aircraft to the fire control system of the M1 tank, from the complex information processing networks of the AEGIS fleet air defense system to the guidance and control system of the MAVERICK air-to-ground missile, whether a tiny chip in PATRIOT or a roomful of stand-alone mainframes in the World Wide Military Command and Control System (WWMCCS), the computer is present. Accompanying this massive infusion of computer hardware into defense systems is, of course, the software which makes it work (67).

Figure 1

ARGUMENT OVERVIEW

Given:

Threat Assessment, National and Service Policies, Strategies, and Plans



Some facts illustrate the increasing importance of software to weapon systems. The Electronic Industries Association estimates DoD expenditures on mission critical software in 1984 to be $9 billion and predicts that it could reach $32 billion by 1990 (1). A review of the FY85 DoD budget presented to Congress by Secretary Weinberger suggests that at least 75% of the programs listed in that review have a software component (2). Among the programs to begin in the 1985-1989 time frame, it can be estimated that at least 80% will have a software development component (3). In addition, over 70% of the technologies, functions, and systems identified in five DoD and Service long range plans will require software (4).

Not only will there be more software in the future, but software will be responsible for more functions. Figure 2 demonstrates the growth in software demand, in millions of object instructions generated, across four generations of the U.S. manned space flight program as increasing numbers of functions were automated. There is a clear trend toward automation through software of more weapon system operations, such as guidance, control, surveillance, intelligence, communication, and navigation (5). Furthermore, the power of each automated function is growing. For example, the current AEGIS air defense system can track in excess of 200 targets while the Ballistic Missile Defense System is planned to track and discriminate tens of thousands of targets (6) - (7).

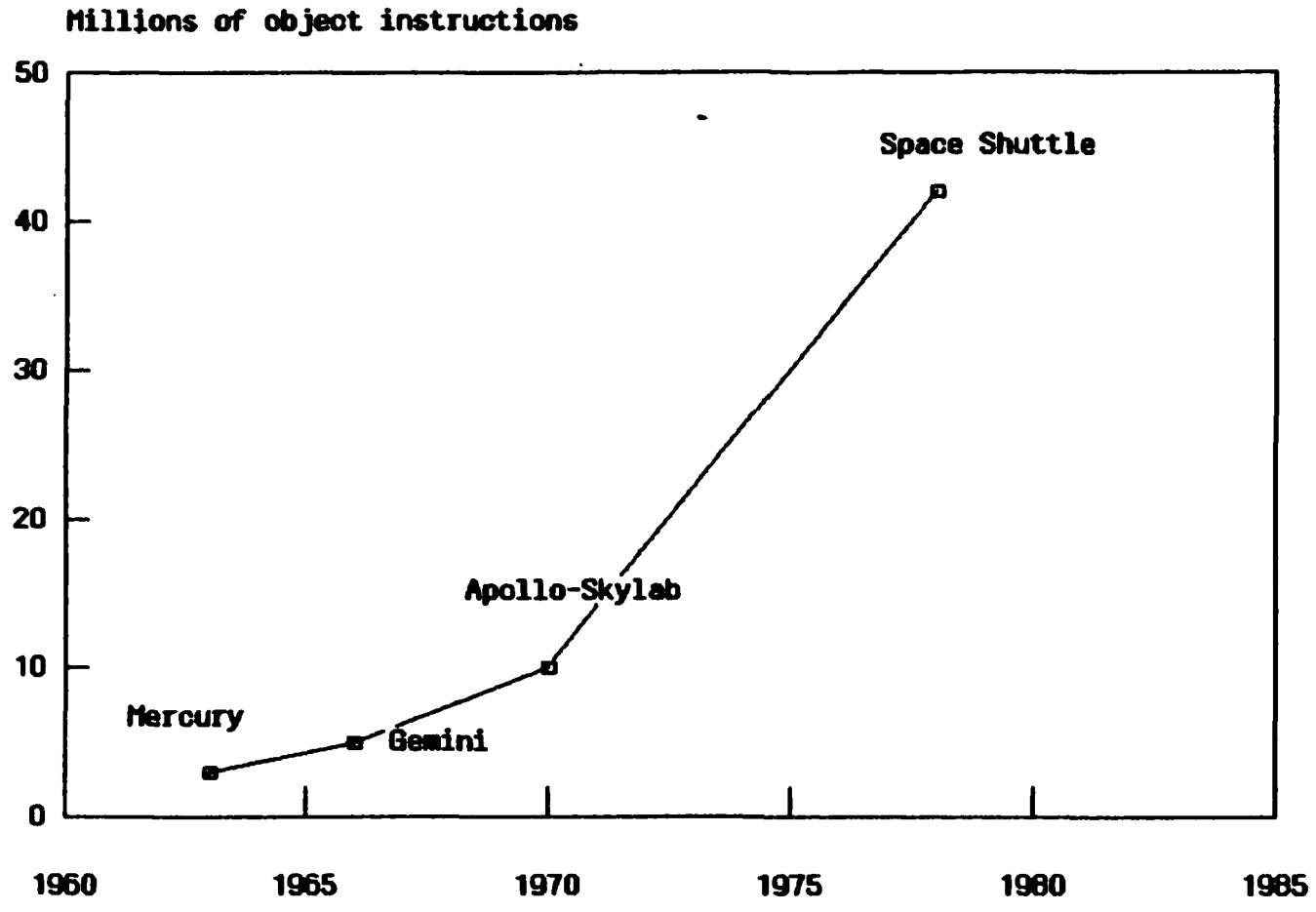
The future will see greater complexity as a result of the trend toward integration of functions for such purposes as sharing of data. The LHX helicopter, currently in the planning stages, illustrates this trend toward integration. Unlike existing helicopters which have no fusion and little information integration capability, the LHX will have multiple sensors integrated into a single display. This information integration capability and other software will enable it to be a single pilot aircraft (8).

Increases in software functionality and complexity imply that software will be increasingly responsible for mission success. Very high reliabilites will therefore be required of future systems. For example, the need has been estimated for the probability of failure in avionics of 10^{-9} per 10 hours flight operation (9).

DoD's software requirements are not the only ones that are increasing as demonstrated by increases in software sales over the past 20 years. During the period from 1963 to 1983, roughly 20 percent of the growth in industry revenues occured in the 16 year period from 1963 to 1979 and 80 percent of the growth occured in the last four years since 1979. From 1981 to 1983 packaged software grew at a 40 percent annual rate. This compared to increases of 26 percent for integrated systems and 16 percent for custom software for that period (142).

Growth in Software Demand

Figure 2



Source: Software Engineering Economics,

by Barry Boehm, Prentice-Hall, 1983

2.2 Manpower Shortage

The total amount of software that the DoD needs to build is well beyond the capability of the number of software engineers that can reasonably be expected to be available, using current software development methods. According to a recent Department of Commerce study (63), the 600,000 to 700,000 programmers and systems analysts in the U.S. have not been able to fill the dramatic increase in demand for their services that resulted from the rapid growth in the use of computers. This increased demand has caused their salaries to rise at an eight to ten percent annual rate since 1978 (63). The current U.S. gap between demand and supply has been estimated in terms of 50,000 to 100,000 software professionals, and if nothing is done, this gap could become 860,000 to 1,000,000 software professionals by 1990 (10) - (11).

During the years 1979 through 1982, employment in software products (packaged software) and professional services (including custom programming) expanded at a 40 percent average annual rate. Even during the recession years of 1980 to 1982, overall software industry employment grew by 17 percent. The software products sector had the highest growth rate of any sector during the 1981-1983 period, increasing threefold from 22,000 to 68,000 employees (63).

The National Science Foundation (NSF) projects an 8.9% - 12.3% annual growth rate for defense requirements for computer specialists as opposed to a 5.4% - 6.4% growth rate for non defense requirements for computer specialists (12). NSF also predicts that the total shortfall for computer specialists will be in the 15 to 30 percent range (about 115,000 to 140,000) by 1987 (12).

The life of some software, especially large military system software can be as long as 15 to 20 years. For these systems, 60 to 75 percent of the costs incurred during the entire life cycle can be attributed to maintenance. Some experts estimate that about 20 percent of this phase consists of error correction and the remaining 80 percent consists of changes and enhancements. In many large organizations that develop their own programs, over one half of staff time may be devoted to these functions, diverting scarce resources from new development (63).

NSF contends that personnel shortages in the computer science area can only be alleviated by large inflows of experienced personnel. Although the computer occupations traditionally have been very flexible in terms of accepting workers from other fields, complex computer applications increasingly demand graduate degrees. Therefore, continued high transfer rates from other occupations may be difficult to sustain, especially as advanced applications are introduced in areas such as CAD/CAM, information technology, telecommunications, and the sophisticated modeling encouraged by development of the supercomputer (13). The shortage

of computer specialists will be even more dramatic as software requirements increase in the future, as is projected (14).

An alternative to alleviating the personnel shortage with large inflows of experienced personnel is to increase productivity. Much of the literature points to a need to increase the productivity of analysts and programmers in the face of these growing workloads and shortages of qualified people (15) - (17).

To date, tools that improve the quality and increase the quantity of software personnel work include high-level languages, software development systems, and application generators. The Department of Commerce report notes, however, that the most progress to date has been made in coding, an area representing only 10 percent of the development effort.

During the 1960's, the annual growth rate in software productivity averaged about 8 to 9% due largely to the transition from assembly language to higher order language software development (128). Higher order languages generally improve productivity by reducing the number of source code lines (129). Programmer productivity may be increased by as much as 5 times when a suitable high-level language is used (132). For example, E.A. Nelson has shown a 3-to-1 productivity improvement for high level language (133).

To date, the adoption of software engineering methods (or modern programming practices) has lagged behind the need for more efficient organization of software production. Similarly, the growth of programmer productivity has been slow in relation to the need for more programs (63).

2.3 Summary of Future Software Technology State of Practice Requirements

Figure 1 outlines this argument as follows. Future MCCR system requirements imply future software requirements that dramatically exceed present requirements and capabilities. Moreover, a growing manpower shortage compounds the problem.

3.0 Software Technology State of Practice (Current)

The current state of practice is having problems meeting current requirements. As a 1985 Air Force Studies Board report (123) notes,

> In spite of numerous past research and development endeavors related to software engineering tools and methodologies, the Air Force continues to experience problems in obtaining required performance, quality, and productivity while controlling cost and schedules of large software based systems (123).

Other sources support the observation that the problems in software development and support are many and varied. They include an inability to project the size or cost of software in DoD programs, deficiencies in project management, and ineffective test and evaluation techniques. These and other problems combine to cause cost escalation and deployment delays. The net result is unnecessary lengthening of the lead time required to meet new enemy threats. Function is reduced and quality suffers as the demand for new systems increases. The final product can be an ineffective or unsuitable system. If improvements are not made, it is reasonable to expect more failures in operational weapon systems in the near future (19).

3.1 General Problems with the State of Practice

Often, weapon system requirements and schedules are not met because of problems with mission critical software. A 1982 DoD task force on software problems observed four major categories of software problems facing the DoD:

- o life cycle
- o production environment
- o product
- o technical and management professionals (20).

The task force subcategorized life cycle problems into problems with:

- o requirements definition and analysis
- o management of life cycle activities
- o software acquisition
- o software product assurance
- o transitions in the life cycle (20).

Support environment problems that the task force identified center around the lack of disciplined methods and development and support tools. Furthermore, it observed that the failure to reuse software results in a high degree of re-invention in software development and higher costs. The task force also stated that capital-investment in support environments has been insufficient (20).

Problems with the software product cited by the task force include the following. The software product does not always meet the need for which it was developed. The lack of good analytical models and hard empirical data on software make it difficult to estimate cost and productivity. Poor designs and inadequate documentation contribute to product-related problems (20).

The task force found problems relating to personnel including an inadequate supply of qualified managers and professionals with a wide range of skills and experience. In addition, present

incentives favor migration of skilled personnel from job to job (20).

A study of the current state of practice on eight software intensive systems (seven government and one commercial) found that five of the development efforts were considered to be successful while three were not. The data gathered does not support the common belief that developments that utilize new technology will succeed when others fail. However, it does point to some common problems with the current state of practice (21).

One observation is that the susceptibility to requirements definition problems is increasing. This is because in each system studied, software was being used to implement functions that had never before been attempted. Further, the capabilities being implemented by software are becoming more varied and critical to mission success. In spite of the importance of the requirements definition process, few systematic techniques and little automated support are used for these activities. The problem of changing requirements compounds other problems found in the current state of practice (21).

In the systems studied, the transition of responsibility for software support from the developing organization (usually commercial) to the Post Deployment Software Support (PDSS) activity (usually governmental) typically did not occur without problems (21). For example, on one of the projects, new tools were developed for post deployment support. It should be noted however, that the tools developed do not support activities specific to the PDSS environment. In fact, it was felt that the same tools would have been beneficial during the initial development. In addition, the original documentation was of poor quality and has been upgraded or, in cases where it did not exist, created (119).

3.2 Problems Estimating the Cost and Size of Software in the DoD

One of the fundamental problems of the current DoD software business is the inability to accurately measure its size. Productivity improvement and future software size estimates have meaning only in comparison to current figures. A 1974 report states that "reliable information on most software and ADP costs in DoD is unavailable in a clearly identifiable form" (22). In 1979, this finding was confirmed by the President's Reorganization Project on Federal Data Processing which said that the total dollar value of computer resources in the DoD is unknown (23).

Isolating software cost data for weapons systems remains a problem in 1984 since a separate line item for software does not exist in the DoD budget. Instead, it is treated as part of individual weapons systems. Major software developments and modifications are part of RDT&E funds, while software maintenance activities are included in Operation and Maintenance funds. Even

DoD program managers sometimes do not know how much is being spent for software on their programs (24).

A 1983 study of 25 U.S. and Japanese organizations comprising some 69 projects found that although nearly all projects collect data, little of it becomes part of corporate memory to be used beyond the project it was collected for. Moreover, in contrast to their Japanese counterparts, U.S. companies rarely use this data for post mortem analyses of projects. In particular, companies experiment with cost and size estimation models, although, at the time the study was done, no one trusted them enough to use them in proposals. Non-standard definitions of types of data are also a problem (18).

The problem of estimating software costs, staffing requirements, and schedules was observed in the study of the current state of practice of eight software intensive systems. The size of the software effort was consistently underestimated and the productivity of the people overestimated (21).

Similarly, the software cost history for an Air Force command and control software project shows that the "best and final offer" on which the winning bidder's contract was obtained was a little over 25% of the original cost expectation, based on "opportunistic assumptions and claims," but the subsequent "series of realizations" resulted in escalating costs and in final costs that were 10 times the winning bid (and almost three times the original cost expectation) (136).

3.3 Project Management Problem

Another chronic problem area in the current state of practice is project management. Several technical weaknesses exist at all levels in the DoD's ability to measure, track, and control software over its life cycle. These weaknesses inhibit the achievement of the management control that is needed over software. A recent survey of software engineering project management problems found that 13 of 20 hypothesized problems actually are important problems that face software engineering project managers. The problems can be broadly classed as planning, organizing, staffing, directing, and controlling problems. The survey found that software engineering project personnel do not generally agree on how to solve these major problems. Moreover, less than 10% of the 100 universities surveyed offer any courses which present a substantial amount of material on software engineeering project management problems although most professors seemed to feel that more classroom time should be allocated to these problems (25). A separate survey found that companies offer little training in the area of project management in comparison to technical education (18).

DoD, industry, and university respondents to a 1982 questionnaire on software technology agreed that the most important software problem is finding and keeping qualified

personnel. This was perceived as much more of a problem than measuring their competence or making best use of them. The respondents also agreed that the second most important problem is poor definition of goals and measures (i.e., software requirements). With few exceptions, all respondents ranked the managerial-related problems in the top half of the problem list (26).

Other problems attributable to project management include the limited use of software standards and tools. Standards vary within a single organization because the software technology group doing data collection, modelling resource usage, and generating standards and practice documents does not have direct authority to enforce adherence to software engineering practices on projects. Software standards are often low and vary across a company because there is no one person at the head of a project organization making software decisions (18).

Likewise, tool use is relatively low across the industry. Tools are most frequently used during the code and unit test phase, in contrast to the requirements or maintenance phases in which fewer tools are used. Corporate management, owing to a limited software background, is not sympathetic to the use of tools. Often a corporate focal point for tool selection, deployment, and evaluation does not exist. Perhaps most importantly, tools are most often bought with project funds forcing projects to adopt both the risk and expenditure for the tool and training. Tool use in Japan is more widespread, mainly because tools are paid for out of overhead, increasing knowledge and use of the tool throughout a company and lowering the risk to any individual project (18).

3.4 Test and Evaluation Problems

Test and evaluation (T&E) is yet another problem area for the current state of practice. The complexity of most defense systems software, coupled with the fact that the software development process in not a routine, regularized, clerical process which can be easily organized, controlled, and graded, leads to a situation in which quantitative approaches to T&E are difficult. The criticality of most software applications makes the quantitative measurement of performance and reliability absolutely necessary (68).

According to a 1983 report of the Software Test and Evaluation Project (STEP), many problems combine to limit the effectiveness of current Defense practices in software test and evaluation. These include: (1) insufficient financial and personnel resources, (2) difficulty in selecting the best testing strategy available for a given application, (3) lack of availability of testing tools, and (4) difficulty of evaluating early development testing since the results of these tests are rarely documented or reported. All of these problems are amplified by frequent modifications to requirements. When budgets

are cut or schedules slip, testing and quality assurance activities are the first casualties.

Among the recommendations made by STEP to improve the current state of practice in software testing are recommendations to develop automated tools for software T&E and recommendations for support of development and basic research which have long-term benefits for software test and evaluation (27).

Another 1983 study of 25 organizations found that all of the projects surveyed did reviews to a greater or lesser extent and agree that they work. Integration testing for the most part consists of stress testing. Among short-term recommendations made by the report were that the review process needs improvement. For the long term, the report recommended the development of a test and evaluation methodology (18).

3.5 Operation Problems

These and other problems with the current state of practice can result in final products that fail to operate correctly. According to a 1981 report, "major U.S. weapon systems are becoming more expensive at an alarming rate and they seldom if ever work well and reliably in operational surroundings" (29). As computers, and software in particular, become increasingly critical components of weapon systems, malfunctions can result in immediate danger, especially in MCCR applications.

Some examples of software problems in the area of defense indicate the potential scope of the problem. The F-18 aircraft had problems with a computer-controlled missile launch. NORAD alerted U.S. forces about incoming Soviet missiles after detecting the moon. A computer glitch caused a Navy warship's three-inch gun to fire in the opposite direction from its intended target (28).

A software bug caused an F-14 to fly off an aircraft carrier into the North Sea (28). Another F-14 was lost to an uncontrollable spin that was traced to tactical software. Yet another F-14 looped as a result of an ill-advised one-line code patch. Fortunately that bug was caught in simulation (121).

The first version of the F-16 navigation software inverted the aircraft whenever it crossed the equator. Fortunately, this was caught in simulation testing (122). The F-16 flight software at one point was unable to right the plane when it was exactly upside down. The program had to choose between rolling right and rolling left and somehow deadlocked (121).

3.6 Summary of Software Technology State of Practice (Current)

There is evidence, therefore, that the state of practice is having problems producing software to meet current requirements. Furthermore, indications are that software is expensive to produce in terms of time, personnel, and financial resources, although it is at present difficult to know the exact cost of software to the DoD. Many problems can be attributed to software management, but there is not general agreement on how to solve such problems and university courses do not devote the time to these problems that professors agree they deserve. Test and evaluation, especially crucial to software reliability, is another area in which the state of practice needs help. Software, as a major component of weapons is increasingly cited as a cause of weapon systems failures.

4.0 Gap Between Current State of Practice and Future State of Practice Requirements

If the current state of practice does not meet current needs, it will not be able to meet future needs either. As Figure 1 shows, there is a gap between the future software technology state of practice requirements and the current software technology state of practice. This gap must be reduced if the U.S. is to see improvement in software development.

The U.S. can close the gap between required software capacity and the current state of practice by exploiting software technology that is now emerging (see section 5.3) and by fostering R&D in new software technologies to meet future requirements. The software factory concept is one example of a state-of-the-art software technology that has shown gains in programmer productivity. However, maturation time for software technology currently averages 15-20 years (66). By accelerating the transition of state-of-the-art software technologies into the state of practice, the DoD can reduce labor intensiveness and latent defects, and improve predictability, performance, and responsiveness to changing threats (46) and (69).

4.1 Japanese Software Factory

One example of a software technology that promises gains in productivity if it is incorporated into the state of practice is the specialized Japanese "Software Factory". Specific productivity and quality figures on Japanese software factory results can be found in (70) - (72).

As an illustration, the productivity rate at Toshiba's Fuchu Works' software factory has increased 14% annually since 1976, reaching an average of 2870 instructions per programmer per month in 1981. Prior to this, from 1972-1976, instructions per programmer per month averaged 1318. This software factory attributes it's high productivity to a return on its investment in

software tools and computer networks, reuse of modules validated through previous applications, and its quality assurance plan (73). Thus, introducing such technologies into the state of practice can improve the ability to produce software.

4.2 Technology Maturation

Research indicates that bringing a technology to the point of maturation where it is popularized and disseminated to a large portion of the technical community generally has taken more than 15-20 years. This includes time for research and concept formulation, development and prototyping, enhancement and exploratory use, and dissemination (74). Software technology maturation case studies upon which this observation is based are (32) - (44). Another study indicates that 4-8 additional years may be required to propagate that technology throughout a large organization (75).

Certain factors that inhibit or facilitate the maturation of technology can be identified and exploited. Particularly important to technology maturation are a recognized need, a receptive target community, and a believable demonstration of cost/benefit. Well-designed channeling of attention and support, an articulate advocate, prior success, incentives, technically astute managers, readily available help, latent demand, simplicity, and incremental extensions to current technology also facilitate the maturation process (76).

For example, technological R&D can be made more productive by understanding and exploiting the S-curve phenomenon. Too often, instead of viewing technological changes in a long-term strategic context and managing it accordingly, managers focus on short-term remedies. This tendency is reinforced by other factos that favor the pursuit of existing technologies and tend to suppress timely exploitation of innovative alternatives (141).

In the earliest phases of a new technology, progress and therefore, R&D productivity is slow. As the technology progresses, so does productivity, reaching a maximum at the midpoint of the curve. It is at this point, before the S-curve flattens out, that managers should be thinking of getting onto a new S-curve where the slope ahead is steeper. Figure 3 illustrates this concept (141).

Making the transition to a new technology is both risky and difficult, but the most common error is poor timing. To preserve long-term strategic flexibility, management should begin to explore technological alternatives when roughly half the full potential of the current technology has yet to be exploited (141).

Unfortunately, common business management tools and practices often mask this phenomenon. Management systems, marketing approaches, and financial analyses tend to reward short-term

MAKING THE TRANSITION TO A NEW TECHNOLOGY

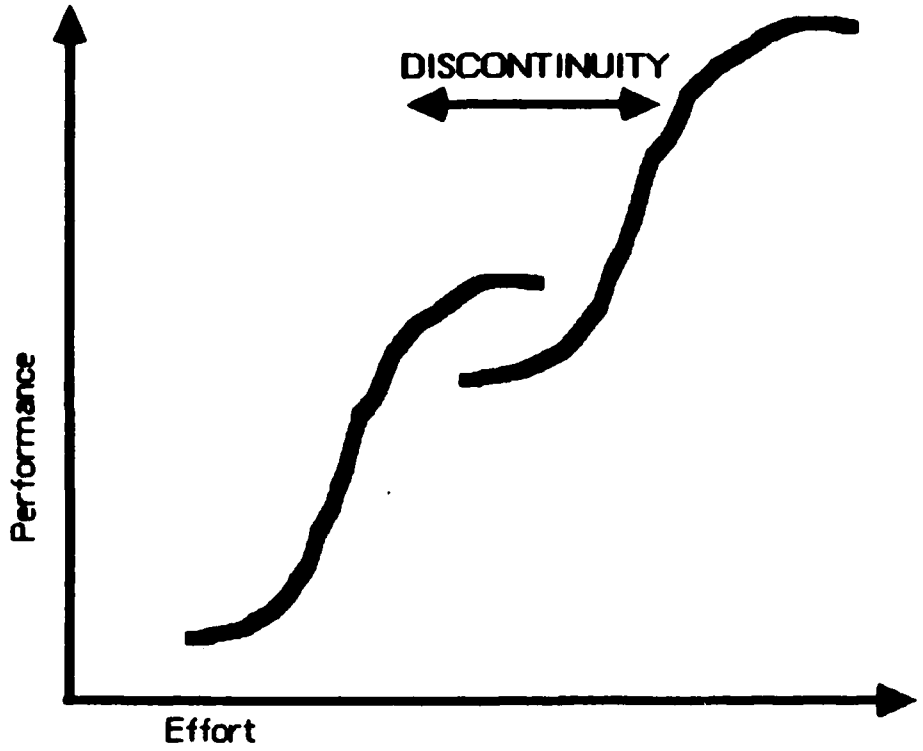


Figure 3

Source: Strategic Management of Technology, by Richard N. Foster, McKinsey Staff Paper, February 1981.

performance based on incremental advances over the competition (141).

Forecasts beyond a 3- or 4-year horizon are usually unreliable, discriminating against a decision to invest in a move to a new technological S-curve where cash flows in early years look less attractive. The portfolio approach to resource allocation tends to emphasize the allocation of resources among existing opportunities, rather than the generation of new ones (141). Thus, the role exists for the U.S. to make the large step to the next S-curve.

This is just one example of how understanding and exploiting the technology maturation process can hasten improvements in the software technology state of the art. Thus the technology maturation process provides the basic context needed for transitioning technology into widespread use.

4.3 Technology Transition

Technology transition activities are planned, overt actions taken to move a piece of technology into widespread use. The challenge is to find ways of introducing new technologies faster in order to produce better software more efficiently. A recent workshop (77) addressed the issue of software engineering technology transfer. Specifically it looked at:

(1) ways of evaluating software technologies and the technology transfer process itself;

(2) training as a technology transfer vehicle;

(3) processes required for effective software engineering technology transfer and the vehicles that might be used to implement such processes; and

(4) behavioral aspects of software technology transfer.

It determined that an important obstacle to technology transfer is the lack of a coherent and well defined software engineering product life cycle process. Common understanding and agreement of a software engineering product life cycle is necessary in order to effectively carry out the technology transfer process. This process consists of:

- identifying available technologies and their costs and benefits;
- overcoming resistance to change;
- selecting appropriate vehicles for technology transfer (depending on the life cycle phase);
- integrating multiple technologies across time and
- overcoming proprietary restrictions.

Some possible transfer vehicles include:

- o education,
- o training,
- o tool provision,
- o personnel transfer,
- o management edict,
- o measurement,
- o consulting,
- o quality circles,
- o documentation,
- o contracts,
- o rewards and incentives,
- o publicity and advertising,
- o demonstrations,
- o experiments,
- o pilot projects,
- o planning mechanisms,
- o competition and challenge,
- o human interface and
- o technical interaction (77).

Thus, the opportunity exists to accelerate the propagation of state-of-the-art software technologies into the state of practice and significantly improve the DoD's capacity to produce defense software.

4.4 Summary of Gap Between Current State of Practice and Future State of Practice Requirements

As evidenced by the Japanese software factory, the potential exists for state-of-the-art software technologies to improve the state of practice. However, based on studies of other software technologies, the natural maturation of a software technology can take 15 to 20 years, too long to improve the state of practice in the near-term. The potential exists to accelerate this process by undertaking technology transition activities.

5.0 STARS Program Fills Gap and Meets Need

Significant basic technology exists and the defense software community is poised to exploit it. A software improvement program is necessary to effect changes in the software state of practice in order to meet future needs. As Figure 1 shows, such a program requires the contributions of DoD activities, industry, and academia as well as unexploited state-of-the-art technologies. Bringing state-of-the-art technologies to the state of practice can help close the gap.

Since the accelerating pace of computing technology is too fast and too capital intensive for any single organization to keep up with alone (78) - (87), a number of DoD and private sector programs have been initiated to improve the software technology state of practice. Articles (78) - (87) strongly imply that

cooperation among Governments, professional societies, and major commercial enterprises and centrally managed financial resources are essential for the survival of industrial nations. A centrally managed effort is therefore necessary to achieve the required capability by the time the DoD needs it.

A centrally managed program and funding source (supplemented by existing defense laboratory programs and industry IR&D (independent research and development) investment), with much of the actual work being contracted through or performed by DoD laboratories and by weapon system programs, would offer the great advantages of elimination of unnecessary duplication and gaps, together with a central focus for identification of opportunities and program advocacy.

5.1 STARS

As a central focal point, the STARS program can help assure that the DoD has the technology it needs, when it needs it, without unnecessary duplication of effort. The STARS program defines, plans, develops, demonstrates, integrates, evaluates, and disseminates tri-service software technology (118).

STARS is specifically designed to mesh closely with and build upon individual Service and defense agency programs, including the Ada program and the Software Engineering Institute (SEI). It will demonstrate and evaluate the feasibility of new techniques such as computer-aided software support systems. STARS will produce mechanisms for implementation and evaluation of other Service agency sponsored software techniques such as the Joint Logistics Commanders' software standards, and the new testing methods resulting from the DoD Software Test and Evaluation Project (STEP) (114).

5.2 DoD, Industry, and University Contributions

Several DoD and private sector programs have been initiated to improve the state of practice of software technology. The DoD initiatives include the STARS and Ada programs, the SEI, DARPA's Strategic Computing Program, the STEP, and the Super Computing Research Center. The Microelectronics and Computer Technology Consortium is addressing the software-related concerns of human interface and early requirements and design. STARS is maintaining close contact with the planners of the still embryonic Software Productivity Consortium to ensure that their efforts are complementary.

The SEI has been established to bridge the gap between R&D activities that demonstrate new techniques and the exploitation of those techniques in system developments in order to effect a significant and rapid improvement in the means of development and support of computer software for mission-critical defense systems. The Software Engineering Study Panel endorsed the creation of the SEI to be a central driver in the technology

insertion process. The SEI was designed to pool scarce resources, provide goal-directed technical management, select high-payoff and mutually-supportive technologies, engineer selected technologies to mission critical scale and quality, and establish visible standards of excellence in practice and ensure that they are met by the whole MCCR software community (88).

A number of the largest defense contractors (and some smaller ones) have made considerable investments in the development of their own internal software engineering environments. With additional funding, it would be possible for them to accelerate the development of these software engineering environments and to tailor them more closely to satisfy DoD needs (89).

5.3 Software Technology State of the Art

The software technology state of art offers opportunities to improve the state of practice. The software engineering environment is one example of a state-of-the-art software technology that is currently receiving a great deal of attention in the literature. In spite of the rapid development of new and potentially superior methods, however, techniques, methods, and practices 10 and 15 years old are routinely used.

5.3.1 Failure to Heed Lessons of the Past

A 1979 study of 50 cases from government, industry, and university software projects in the Los Angeles area estimates that lessons learned and published 18 years earlier in software engineering are not heeded half of the time. Some factors that account for this lag include rapid technological change, education shortfalls, technology transition inhibitions, resistance to disciplined methods, inappropriate role models, and a restricted view of software engineering (30).

5.3.2 Failure to Use New Methods

A study of seven software-intensive government systems and one commercial system found that technologies applied during software development can usually be traced to the requirements of the military standards that have been referenced in the contracts. In general, developers are not rewarded for exceeding the minimum technological requirements of the standards, and these developers did not do so. Relatively few automated tools were available or used by any of the projects for design, coding, testing, or software support (31).

A 1983 survey of several companies and organizations describes the software engineering technologies being used for projects of different types. For the most part, the industry at large rarely uses software engineering technologies correctly. General characteristics of the software development environment are that

- o requirements are most often done in natural language text with little tool support;
- o there is little traceability between specifications and design;
- o designs are expressed most often in some form of Program Design Language (PDL)
- o although most coding is done in high level languages, it lacks proper machine support;
- o the integration test phase consists mostly of stress testing; and
- o maintenance is rarely performed by the software developer.

The report also observed that most companies are willing to invest in hardware but not software. Problems with the current technology include the inability to transport tools among hardware and the failure of many tools to live up to promises (18).

5.3.3 New Methods Do Exist

Substantial unexploited software technology exists. Some state-of-the-art technologies offer the ability to correct many of the underlying causes of the problems described above, but are not widely used (45), (113), (115) - (117). Concrete examples of some of these unexploited software technologies are found in (65).

A 1982 document (45), drawing from several Defense Science Board recommendations, earlier planning documents, and independent recommendations proposed by interested and concerned groups, identified and assessed 13 areas with substantial opportunities for improving the software technology state of practice:

- o integrated support environments,
- o system definition technologies,
- o maintenance techniques,
- o reliability technologies,
- o database technologies,
- o distributed systems,
- o knowledge-based systems,
- o hardware/software synergy,
- o human factors,
- o technology transfer,
- o measurement technologies,
- o management tools and techniques and
- o applications-oriented technologies and software reuse (45).

A 1983 conference on software development tools, techniques, and alternatives identified an extensive list of state-of-the-art technologies with the potential to improve the state of the practice including:

- o object oriented programming,
- o software management tools,
- o rapid prototyping and application generators,

- debugging tools,
- software design tools,
- software development environments and
- software development data management (46).

A 1983 report on the software state of practice made some short-term and long-term recommendations for improvement. Short-term recommendations included:

- making available more and better computer resources for development;
- evaluating methods and tools;
- building tool support for a common high level language;
- improving review processes;
- using incremental development; and
- collecting and analyzing data.

For the long-term, the report recommended:

- maintaining compiler technology;
- trying prototyping;
- developing test and evaluation tools;
- examining the maintenance process and
- encouraging innovation (18).

A 1983 Air Force Scientific Advisory Board Ad Hoc Committee on Software recommended establishing a Software Engineering and Computer System Technology and Support Center to collect and focus Air Force resources on software issues. It contends that the Air Force must have a central organization to concentrate and focus the R&D expenditures required to make significant improvements in the underlying technology essential to providing the needed technology improvements in the software acquisition and production process. In addition, it recommends increasing investment for software engineering tooling and increasing funding for long range software production technology research to insure the acceleration of advanced technologies. Such technologies include: very high order languages, applications generators, knowledge-based systems, reuse of application units, and verification technology (64).

For example, it has been estimated that Ada will reduce coding effort in the far-term by 44% for applications software development and 40% for support software. Estimates for reduction in lines of code from Ada range from 5 to 10% in the near-term to 50% in the long-term (130). Another source hypothesizes a 75% improvement in coding efficiency from the continued spread of new and powerful languages such as Pascal and (possibly) Ada (131). Another way in which productivity of coding might be improved in the "near" future is finding and correcting errors in groups early on. This has been shown to translate to a 23% increase in productivity of coding (134).

Application generators and formal specification and transformation systems also show promise. Application generators

are software packages designed to help end-users build applications in a given domain. Formal specification and transformation systems use a formal specification language to express system design. The resulting specification is automatically checked for consistency and completeness after which it is transformed into an executable program either by automatic means or by interaction with the designer (135).

A 1985 Air Force Studies Board report on methods to improve software quality and life cycle cost made several recommendations. These included:

- o Develop first, second, and third generation software engineering environments to incorporate increasing levels of comprehensive and integrated tools and techniques such as Artificial Intelligence;
- o Improve management of software in the acquisition, development, and operations and maintenance phases by
  - increasing use of prototyping,
  - creating centers of expertise in software,
  - developing software management tools,
  - providing better training and incentives, and
  - improving procedures for selecting managers.
- o Promote resuable support tools, models, subsystems, and Ada packages (123).

Careful management approaches to computer development and the development of more sophisticated man-machine interfaces will improve productivity. Prototyping and reusable code and design are also attractive options (135).

Although the level of detail varies in the lists of opportunities presented above, there is overlap between them. This indicates agreement on certain candidate technologies to improve the state of practice. One such technology that has received intensive attention over the past few years is the software engineering environment (47) - (51). The Japanese, British, European Community, Brazilians, and many U.S. commercial enterprises are embarking on the development of such systems. The time appears ripe for exploitation of this technology (52) - (60).

The Japanese environment project, known as Sigma, will develop a Japan-wide Unix-based network. It will offer four software engineering databases for software modules, software tools, software products, and technical information (124). The Alvey Directorate in the United Kingdom is developing three generations of Integrated Project Support Environments. The first generation will be a file based Unix system, the second will be database-oriented with a distributed operating system, and the third will be intelligent (knowledge based system) oriented (125). Brazil has plans to develop a software factory (126).

The European Economic Community has recently made a $1 billion commitment to the European Strategic Program for Research and Development in Information Technology (ESPRIT) (61). ESPRIT pools the research efforts of a dozen European electronics firms concerned with several technologies, including software. The ESPRIT Project is funded jointly by the European governments with an equal amount being added by the companies (62).

ESPRIT is working on the development of a Portable Common Tool Environment (PCTE) which will provide a common base for software engineering environments aimed at supporting European cooperative research and development and at fostering widespread dissemination and exploitation of emerging technologies, methods, and tools. For its wide availability, Unix is expected to play the role of "initial common environment" for the time being (127). To date, PCTE specifications have been developed.

5.4 Summary STARS Program Fills Gap and Meets Need

Given that substantial unexploited software technology exists and that software problems and requirements have much in common across the Services, a centrally managed DoD software improvement program can improve the state of practice of software technology. Ada and the Joint Logistics Commanders' software standards, by fostering uniformity and cooperation among the Services have prepared the DoD community sociologically and organizationally for change. The time is now right for a centrally-managed DoD effort such as STARS to exploit this uniformity among the Services in order to improve productivity and to achieve greater system reliability and adaptability, while at the same time, coordinating the efforts of the private sector with those of the DoD laboratories and organizations like the SEI to stimulate R&D and technology insertion.

STARS will provide the means by which laboratories and program managers, and their industry counterparts will have access to usable technology products vital to future mission critical systems. The SEI, as a partner to STARS will encourage and accelerate the transition of these technology products into practice with respect to DoD mission critical systems (114). Their use could also aid U.S. international competitiveness.

6.0 Summary of the Arguments

Software for future MCCR systems will be more pervasive and perform more functions than current software. Consequently, it will be more complex and critical to mission success. Moreover, there is a growing shortage of computer personnel to meet these needs. The future software state of practice will therefore have to plan and build many larger, more complex, more reliable, and more maintainable systems less labor intensively than the current state of practice does today. The result is a gap between the

future state of practice requirements and the current state of practice.

To compound the problem, there is evidence that the current state of practice has trouble meeting current requirements. Often, state-of-the-art technologies that could be used are not. This is because widespread popularization of a software technology can take more than 15-20 years. The opportunity exists to reduce this lag by undertaking activities that will facilitate rather than inhibit technology transition. Thus, as Figure 1 indicates, accelerating transition of the state of the art into the state of practice can be used to close the technology insertion gap.

In those areas where state-of-the-art technologies do not exist to improve the state of practice, R&D must be fostered. DoD activities, industry, and academia are making contributions in this area, but the total effort is too big an undertaking for any one entity.

Therefore, the STARS program is necessary to accelerate, coordinate and disseminate the results of R&D in software technology. As Figure 1 shows, STARS bridges the gulf between future and current software technology states of practice, meeting the need for an improved software state of practice.

REFERENCES

1. DoD Digital Data Processing Study - A Ten Year Forecast, Electronic Industries Association, 1980, p. 120.

2. Redwine, Samuel T., Jr. et al, DoD Related Software Technology Requirements, Practices, and Prospects for the Future, Institute for Defense Analyses, IDA Paper P-1788, June 1984, p. 2.

3. Ibid., p. 13.

4. Ibid., p. 16.

5. Probert, Thomas H. et al, An Assessment of Software Technology in Defense Systems (Interim Report), Institute for Defense Analyses, March 1984 - no page numbers.

6. Op Cit., Redwine, Samuel T., Jr., et al DoD Related Software Technology Requirements, Practices, and Prospects for the Future, p. 25.

7. Report of the Study on Eliminating the Threat Posed by Nuclear Ballistic Missiles, Vol. V (Battle Management, Communications, and Data Processing), Defensive Technologies Study Team, October 1983.

8. Op Cit., DoD Related Software Technology Requirements, Practices, and Prospects for the Future, p. 24.

9. Murray, N. et al, "Highly Reliable Multiprocessors," AGARDograph No. 224, Integrity in Electronic Flight Control Systems, edited by P.R. Kurzhals, Advisory Group for Aerospace Research and Development, April 1977, pp. 17.1-17.6.

10. Martin, E.W., "Strategy for a DoD Software Initiative," Computer, (March 1983), p. 53.

11. "Software Technology in the 1990's Using the Current Life Cycle Paradigm," Appendix IIIA, in Strategy for a DoD Software Initiative, Volume II: Appendices, Department of Defense, 1 October 1982, p. 207.

12. Projected Response of the Science, Engineering, and Technical Labor Market to Defense and Non-Defense Needs: 1982-87, National Science Foundation, NSF 84-304, January 1984, pp. 36-39.

13. Labor-Market Conditions for Engineers: Is There a Shortage?: Proceedings of a Conference, Office of Scientific and Engineering Personnel, National Research Council, National Academy Press, June 1984, p. 56.

14. Rakoczki, L., "The Software Liberation Party, Break Away from the Programmer's 'Personal Touch,'" Data Management, volume 17, number 4, April 1979, pp. 14-16.

15. LaBelle, Charles et al, Finding, Selecting, Developing, and Retaining Data Processing Professionals Through Effective Human Resources Management, Van Nostrand Reinhold Company, 1983.

16. Bosworth, G.H., "Managing a Programmer Shortage," Datamation, volume 27, number 9, 25 August 1981, pp. 82 and 86.

17. Op Cit., Labor Market Conditions for Engineers: Is there a Shortage?.

18. Zelkowitz, M.V., et al, The Software Industry: State of the Art Survey, Technical Report, TR-1290, Department of Computer Science, University of Maryland, May 1983.

19. Report of the DoD Joint Service Task Force on Software Problems, Department of Defense, July 30, 1982, pp. 28-29.

20. Ibid., pp. 10-27.

21. Op Cit., Redwine, Samuel T., Jr., et al, DoD Related Software Technology Requirements, Practices, and Prospects for the Future, pp. 74-79.

22. Fisher, David A., Automatic Data Processing Costs in the Defense Department, Institute for Defense Analyses, IDA Paper P-1046, October 1974, p.2.

23. Jensen, Alton P. et al, Information Technology and Governmental Reorganization: Summary of the Federal Data Processing Reorganization Project, Office of Management and Budget, President's Reorganization Project, April 1979.

24. Op Cit., Redwine, Samuel T., Jr., et al, DoD Related Software Technology Requirements, Practices, and Prospects for the Future, p. A-3).

25. Thayer, R.H., et al, "Major Issues in Software Engineering Project Management," IEEE Transactions on Software Engineering, volume SE-7, number 4, pp. 333-42, July 1981.

26. Siegel, Eric D., Summary of Responses to the Software Technology Initiative Questionnaire, Mitre Corporation, MTR-82w00085, May 1982, pp. 6-8.

27. DeMillo, R.A. and Martin, R.J., OSD/DDT&E Software Test and Evaluation Project, Volume 1, Report and Recommendations, School of Information and Computer Science Georgia Institute of Technology, Atlanta, GA, pp. 33-34.

28. Neumann, Peter G., "Letter from the Editor," Software Engineering Notes, volume 8, no. 5, October 1983, pp. 1-6.

29. Heatherly, Charles L., editor, Mandate for Leadership: Policy Management in a Conservative Administration, The Heritage Foundation, 1981, p. 131.

30. Boehm, B.W., "Software Engineering-As it Is," Proceedings of the 4th International Conference on Software Engineering 11-21, 1979.

31. Op Cit., Redwine, Samuel T., Jr., et al, DoD Related Software Technology Requirements, Practices, and Prospects for the Future, p. 77.

32. Bailey, John, Cost Model Technology Transition, May 1984.

33. Clements, Paul C., et al, Case Studies of Software Engineering Technology Transfer, Tech. Memorandum, Naval Research Laboratory, April 1984.

34. DeMillo, Richard A., Compiler Technology Insertion Network Study, May 1984.

35. Manley, John H., Technology Case Study: Software Engineering Concepts, Tech. Memo, Computing Technology Transition, Inc., Madison, Connecticut, May 1984.

36. Manley, John H., Technology Case Study: Software Metrics, Tech. Memo, Computing Technology Transition, Inc., Madison, Connecticut, April 1984.

37. Manley, John H., AFR 800-14 History, Tech. Memo, Computing Technology Transition, Inc., Madison, Connecticut, May 1984.

38. Marmor-Squires, Ann, Formal Software Verification as an Example of Software Technology Transfer, TRW Defense Systems Group, Fairfax, VA, May 1984.

39. Martin, R.J., DOD-STD-SDS: The Development of a Standard, May 1984.

40. Redwine, Samuel T., Structured Programming: A Technology Insertion Case Study, Computer and Software Engineering Division, Institute for Defense Analyses, May 1984.

41. Riddle, William E., "The Magic Number Eighteen Plus or Minus Three: A Study of Software Technology Maturation," ACM SIGSOFT Software Engineering Notes, 9, 2 (April 1984) (Includes case studies of Unix, Smalltalk-80, and SREM).

42. Riddle, William E., Knowledge-based Systems as a Case Study in Software Technology Maturation, SDAM/15, software design & analysis, inc., April 1984.

43. Riddle, William E., Abstract Data Types as a Case Study in Software Technology Maturation, SDAM/16, software design & analysis, inc., April 1984.

44. Weiss, David, Time Line for Development and Transfer of SCR Methodology, February 1984.

45. "Appendix II - Opportunity Assessments," Strategy for a DoD Software Initiative, Volume II: Appendices, Department of Defense, 1 October 1982, pp. 11-200.

46. Softfair, A Conference on Software Development Tools, Techniques, and Alternatives, Arlington, VA Jul 25-28, 1983, pp. 267-395.

47. Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Environments, Symposium on Practical Software Development Environments, Pittsburgh, PA, April 23-25, edited by Peter Henderson, in Software Engineering Notes, volume 9, number 3, May 1984.

48. Interactive Programming Environments, edited by David R. Barstow, Howard E. Shrobe and Erik Sandewall, New York, McGraw-Hill, 1984.

49. Software Engineering Environments, Proceedings of the Symposium held in Lahnstein, Federal Republic of Germany, June 16-20, 1980 edited by Horst Hunke, North-Holland, 1981.

50. IEEE Computer Society 1984 Conference on Ada Applications and Environments, October 15-18, 1984, St. Paul, MN, sponsored by IEEE-CS Computer Languages Technical Committee.

51. Proceedings of the Symposium on Application and Assessment of Automated Tools for Software Development, November 1-3, 1983, San Francisco, CA, sponsored by IEEE Computer Society and University of Texas at Austin.

52. "Software Engineering: The Age of the Software Factory is Here," Special Issue, Computer, volume 8, number 5, May 1975.

53. Bratman, H. and T. Court, "The Software Factory, Computer, volume 8, number 5, May 1975, pp. 28-37.

54. McNamara, D.M., "Japanese Software Factories," A Public Briefing, General Electric Corporate Information Systems, Bridgeport, Connecticut, 1983.

55. Kim, K.H., "A Look at Japan's Development of Software Engineering Technology," Computer, volume 16, number 5, May 1983, pp. 26-37.

56. Martin, E., "Keynote Address," Proceedings Milcom III - Military Computers and Software, American Defense Preparedness Association, Washington, D.C., 25-26 January 1984, pp. 3-19.

57. Manley, J.H., "The Emerging Technologies - The STARS Program," Proceedings Milcom III - Military Computers and Software, American Defense Preparedness Association, Washington, D.C., 25-26 January 1984, pp. 46-48.

58. Tajima, D. and T. Matsubara, "Inside the Japanese Software Factory," Computer, volume 17, number 3, March 1984, pp. 34-43.

59. Zelkowitz, M.V., et al, "Software Engineering Practices in the U.S. and Japan," Computer, volume 17, number 6, June 1984, pp. 57-66.

60. Manley, J.H., "Computer Aided Software Engineering (CASE) Foundation for Software Factories," Proceedings COMPCON Fall '84.

61. "International Report," Computerworld, February 4, 1985, p. 24.

62. DeMillo, Richard A., et al, Software Engineering Environments for Mission Critical Applications--STARS ALternative Programmatic Approaches, Institute for Defense Analyses, IDA Paper P-1789, August 1984, p. A-66.

63. A Competitive Assessment of the U.S. Software Industry, Science and Electronics, Office of Computers and Business Equipment, Assistant Secretary for Trade Development, Department of Commerce, December 1984.

64. Report of the U.S. Air Force Scientific Advisory Board Ad Hoc Committee on the High Cost and Risk of Mission-Critical Software, U.S. Air Force Scientific Advisory Board Ad Hoc Committee on the High Cost and Risk of Mission-Critical Software, December 1983, pp. 4-1 - 4-5.

65. Op. Cit. Redwine, Samuel T., Jr. et al, DoD Related Software Technology Requirements, Practices, and Prospects for the Future, p. 110.

66. Redwine, Samuel T. and Riddle, William E., "Software Technology Maturation," 8th International Conference on Software Engineering, August 1985.

67. Linder, Isham (RAdm Ret., USN), "Keynote Address," NSIA/DoD National Conference on Software Test and Evaluation, 1-3 February, 1983, p. 2.

68. Ibid., p. 3.

69. Op Cit., Report of the U.S. Air Force Scientific Advisory Board Ad Hoc Committee on the High Cost and Risk of Mission Critical Software.

70. Namara, D.M., "Japanese Software Factories," A Public Briefing, General Electric Corporate Information Systems, Bridgeport, Connecticut, 1983.

71. Kim, K.H., "A Look at Japan's Development of Software Engineering Technology," Computer, volume 16, number 5, May 1983, pp. 26-37.

72. Manley, J.H., "Computer Aided Software Engineering (CASE) Foundation for Software Factories," Proceedings COMPCON Fall '84.

73. Op Cit., Kim, K.H., "A Look at Japan's Development of Software Engineering Technology," p. 32.

74. Op Cit., Redwine, Samuel T., Jr. et al, DoD Related Software Technology Requirements, Practices, and Prospects for the Future, p. 137.

75. Willis, R.R., "Technology Transfer Takes 6 Plus/Minus 2 Year," Proceedings IEEE Workshop on Software Engineering Technology Transfer, April 1983.

76. Op Cit., Redwine, Samuel T., Jr. et al, DoD Related Software Technology Requirements, Practices, and Prospects for the Future, pp. 94-100.

77. IEEE Computer Society Workshop on Software Engineering Technology Transfer, Miami Beach, Florida, April 25-27, 1983.

78. A Programme for Advanced Information Technology, The Report of the Alvey Committee, Department of Industry, London, England, Her Majesty's Stationery Office, October 1982.

79. Karatsu, H., Fifth Generation Computer Systems, North-Holland Publishing Company, Amsterdam,-New York-Oxford, 1982.

80. Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, Department of Defense, Office of the Secretary of Defense, OUSDRE (R&AT)-CSS, 15 March 1983.

81. Musa, J.D., Ed., "Stimulating Software Engineering Progress - A Report of the Software Engineering Planning Group," ACM Software Engineering Notes, volume 8, number 2, April 1983, pp. 29-54.

82. Steier, R., Ed., "Cooperation is the Key: An Interview with B.R. Inman," Communications of the ACM, volume 26, number 9, September 1983, pp. 642-645.

83. Withington, F.G., "Winners and Losers in the Fifth Generation," Datamation, volume 29, number 12, November 1983, pp. 193-209.

84. Sumney, L., "The Evolving Partnership Congress/Industry/Military - Industry Cooperation," Proceedings Milcom III - Military Computers and Software, American Defense Preparedness Association, Washington, D.C., 25-26 January 1984, pp. 46-48.

85. Heffernan, H., "DoD to Set Up $74 Million Software Institute," Government Computer News, volume 3, no 4, April 1984.

86. "Reshaping the Computer Industry," Business Week, July 16, 1984, pp. 84-111.

87. "Focus on Japan," Special Issue of Creative Computing, volume 10, number 8, August 1984.

88. Report of Findings and Recommendations -- Software Engineering Institute Study Panel, IDA Record Document D-49, Institute for Defense Analyses, December 1983, p. 9.

89. Private communication with Vance Mall.

90. Druffel, Larry E., Samuel T. Redwine, Jr., and William E. Riddle, "Guest Editors' Introduction: The DoD STARS Program," Computer, volume 16, number 11, November 1983, pp. 9-13.

91. Martin, Edith W., "The Context of STARS," Computer, volume 16, number 11, November 1983, pp. 14-20.

92. Druffel, Larry E., Samuel T. Redwine, Jr., and William E. Riddle, "The STARS Program: Overview and Rationale," Computer, volume 16, number 11, November 1983, pp. 21-29.

93. Boehm, Barry W., and Thomas A. Standish, "Software Technology in the 1990's: Using an Evolutionary Paradigm," Computer, volume 16, number 11, November 1983, pp. 30-38.

94. Balzer, Robert, Thomas E. Cheatham, Jr., and Cordell Green, "Software Technology in the 1990's: Using a New Paradigm," Computer, volume 16, number 11, November 1983, pp. 39-46.

95. Dunham, Janet R., and Elizabeth Kruesi, "The Measurement Task Area," Computer, volume 16, number 11, November 1983, pp. 47-55.

96. Lubbes, H.O., "The Project Management Task Area," Computer, volume 16, number 11, November 1983, pp. 56-66.

97. Oglesby, Charles E., and Joseph E. Urban, "The Human Resources Task Area," Computer, volume 16, number 11, November 1983, pp. 65-70.

98. Frank, Geoffrey A., Samuel T. Redwine, Jr., and Stephen L. Squires, "The Systems Task Area," Computer, volume 16, number 11, November 1983, pp. 71-77.

99. Batz, Joseph C., Paul M. Cohen, Samuel T. Redwine, Jr., and John R. Rice, "The Application-Specific Task Area," Computer, volume 16, number 11, November 1983, pp. 78-85.

100. Kruesi, Elizabeth, "The Human Engineering Task Area," Computer, volume 16, number 11, November 1983, pp. 86-96.

101. Marmor-Squires, Ann B., William E. Riddle, George E. Sumrall, and Jack C. Wileden, "The Support Systems Task Area," Computer, volume 15, number 11, November 1983, pp. 97-103.

102. "Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy," DoD, 15 March 1983 in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

103. "Software Technology for Adaptable, Reliable Systems (STARS) Functional Task Area Strategy for Measurement," DoD, 30 March 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

104. "Software Technology for Adaptable, Reliable Systems (STARS) Functional Task Area Strategy for Human Resources," DoD, 30 March 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

105. "Software Technology for Adaptable, Reliable Systems (STARS) Functional Task Area Strategy for Project Management," DoD, 30 March 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

106. "Software Technology for Adaptable, Reliable Systems (STARS) Functional Task Area Strategy for Systems," DoD, 21 April 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

107. "Software Technology for Adaptable, Reliable Systems (STARS) Functional Task Area Strategy for Application Specific," DoD, 30 March 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

108. "Software Technology for Adaptable, Reliable Systems (STARS) Functional Task Area Strategy for Acquisition," DoD, 30 March 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

109. "Software Technology for Adaptable, Reliable Systems (STARS) Functional Task Area Strategy for Human Engineering," DoD, 30 March 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

110."Software Technology for Adaptable, Reliable Systems (STARS) Functional Task Area Strategy for Support Systems," DoD,. 30 March 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

111. "A Candidate Strategy for the Software Engineering Institute," March 15, 1983, in Software Technology for Adaptable, Reliable Systems (STARS) Program Strategy, 1 April 1983.

112.Strategy for a DoD Software Initiative, DoD, 1 October 1982.

113.Samuel T. Redwine, Jr., Eric D. Siegel, and Gilbert R. Berglass, Candidate R&D Thrusts for the Software Technology Initiative, DoD, Prepared with the Assistance of the MITRE Corporation, May 1981.

114.Letter to the Honorable Melvin Price, Chairman, Committee on Armed Services, U.S. House of Representatives from the Honorable Richard D. DeLauer, Under Secretary of Defense for Research and Engineering, responding to questions raised by House and Senate reports on the Department of Defense Appropriations Bill for 1984, 17 September 1984.

115.Computer Technology Forecast and Weapon Systems Impact Study (COMTEC-2000), 3 volumes, COMTEC 2000 Study Group, HQ Air Force Systems Command, Technical Rept. 78-03, December 1978 - July 1979.

116.Wegner, P. (editor), Research Directions in Software Technology, MIT Press, 1979.

117.Arden, B. (editor), What Can Be Automated?, MIT Press, 1980.

118.Software Technology for Adaptable, Reliable Systems (STARS) Program Summary, Computer Software and Systems Directorate, Deputy Under Secretary of Defense (Research and Advanced Technology), September 1984.

119.Op. Cit. Redwine, Samuel T., Jr. et al, DoD Related Software Technology Requirements, Practices, and Prospects for the Future, p. 75.

120.Op Cit. Edith W. Martin, "The Context of STARS," p. 15.

121.Neumann, Peter G., "Letter from the Editor," Software Engineering Notes, volume 9, number 5, October 1984, p. 2.

122.Neumann, Peter G., "Letter from the Editor," Software Engineering Notes, volume 5, number 2, April 1984, p. 6.

123. Methods for Improving Software Quality and Life Cycle Cost, Committee on Methods for Improving Software Quality and Life Cycle Cost, Air Force Studies Board, Commission on Engineering and

Technical Systems, National Research Council, National Academy Press, 1985, pp. 5-8.

124. "News- International Report - Japan," Computerworld, May 20, 1985, p. 20.

125. Brian Oakley, Director, Alvey, Presentation given at the STARS Industry and NSIA STARS Conference, 29 April - 2 May, 1985.

126. Per telephone conversation with Joel Trimble, Computer Software and Systems Directorate, OUSDRE, DoD, 694-0208.

127. "Council Decision of 11 February 1985 Adopting the 1985 Work Programme for the European Strategic Programme for Research and Development in Information Technologies: ESPRIT," Official Journal of the European Communities, volume 28, L55, February 1985, p. 38.

128. Boehm, Barry, Software Engineering Economics, Prentice-Hall, 1983, p. 644.

129. Bernstein, Lawrence and Yuhas, Christine M., "Blood from Turnips?", Datamation, January 15, 1985, p. 110.

130. Douville, Anne A., Salasin, John, and Probert, Thomas H., The Impact of Ada on COCOMO Cost Estimates as Applied to the World Wide Military Command and Control (WWMCCS) Information System (WIS), Institute for Defense Analyses, IDA Paper P-1810, January 1985, pp. S-2 and S-5.

131. Horowitz, Ellis and Munson, John, "An Expansive View of Reusable Software," IEEE Transactions on Software Engineering, vol. SE-10, no. 5, September 1984, p. 478.

132. Brooks, Frederick P., Jr., "The Mythical Man-Month," Datamation, December 1974.

133. Nelson, E.A., Management Handbook for the Estimation of Computer Programming Costs, Report TM-3225, System Development Corporation.

134. Fagan, M.E., "Design and Code Inspections to Reduce Errors in Program Development," IBM Systems Journal, vol. 15, no. 3, 1976.

135. Op cit., Horowitz, Ellis and Munson, John, "An Expansive View of Reusable Software," p. 479.

136. Op cit., Report of the U.S. Air Force Scientific Advisory Board Ad Hoc Committee on the High Cost and Risk of Mission-Critical Software, pp. 3-9 - 3-10.

137. 1983 Summer Study on Acquiring Software, Army Science Board, 1983.

138. STARS Software Technology for Adaptable Reliable Systems, Defense Industry Briefing, STARS Joint Program Office, April 29, 1985.

139. "What Will We Buy with Fifty-Two Million Dollars in STARS in FY 1986?," STARS Joint Program Office Answer to Congressional Question, March 1985.

140. Department of Defense Computer Technology (Study Annex) A Report to Congress, Office of the Under Secretary of Defense Research and Engineering, Washington, DC, January 1984.

141. Foster, Richard N., Strategic Management of Technology, McKinsey Staff Paper, February 1981.

142. Op cit., A Competitive Assessment of the U.S. Software Industry, p. 18.

Distribution List for IDA Paper P-1872

DoD

Col. Joe Greene — 10 copies
Director, STARS Joint Program Office
1211 Fern St., C-107
Arlington, VA 22202

Defense Technical Information Center — 2 copies
Cameron Station
Alexandria, VA 22314

CSED Review Panel

Dr. Dan Alpert, Director
Center for Advanced Study
University of Illinois
912 W. Illinois Street
Urbana, Illinois 61801

Dr. Barry W. Boehm
TRW Defense Systems Group
MS 2-2304
One Space Park
Redondo Beach, CA 90278

Dr. Ruth Davis
The Pymatuning Group, Inc.
2000 N. 15th Street, Suite 707
Arlington, VA 22201

Dr. Larry E. Druffel
Software Engineering Institute
Shadyside Place
480 South Aiken Av.
Pittsburgh, PA 15231

Dr. C.E. Hutchinson, Dean
Thayer School of Engineering
Dartmouth College
Hanover, NH 03755

Mr. A.J. Jordano
Manager, Systems & Software
Engineering Headquarters
Federal Systems Division
6600 Rockledge Dr.
Bethesda, MD 20817

Mr. Robert K. Lehto
Mainstay
302 Mill St.
Occoquan, VA 22125

Mr. Oliver Selfridge
45 Percy Road
Lexington, MA 02173

IDA

Gen. W.Y. Smith, HQ
Mr. Seymour Deitchman, HQ
Ms. Karen Webber, HQ
Dr. Jack Kramer, CSED
Dr. John Salasin, CSED
Dr. Robert Winner, CSED
Ms. Katydean Price, CSED 2 copies
IDA C&D Vault 3 copies

END

12-87

DTIC